Remarks

Reconsideration of the above-captioned application is respectfully requested. Claims 6, 7, 11, and 13 have been rejected under 35 U.S.C. §102 as being anticipated by Shimuzu et al., USPN 6,772,343, while Claims 8-10, 12, 14, and 15-17 been rejected under 35 U.S.C. §103 as being obvious over Shimuzu et al.

Claim Summary

By way of explanation of the invention without implying any claim limitations by reference to the specification, Claim 6 sets forth a computer program device (element 16, page 6, line 4, figure 1) that includes a computer program storage device including a program of instructions usable by an encryption computer (14, id.). Included are logic means for chaining a data block to a plain text version of an adjacent block in the stream to render a chained block, figure 2, page 7, last paragraph. Also included are logic means for scrambling the chained block using a first round of a cipher to render a scrambled block, id. Also, logic means are provided for iterating the means for scrambling and chaining using subsequent rounds of the cipher, page 8, first paragraph.

Claim 8 is more detailed, reciting a computer system for encrypting a stream of data blocks that includes a processor (14, figure 1) that is programmed to receive a sequence of N blocks and to initialize a previous block variable B, page 7, penultimate paragraph, figure 2. Also, for i=1 to N, a loop is executed that includes XORing an ith block with B to render a modified ith block, setting B equal to the modified ith block, scrambling the modified ith block using at least one round of a cipher, and incrementing "i" by unity and returning, figure 2, page 7, last paragraph. A previous block variable B is initialized and then, for i=N to 1, a DO loop is executed to XOR an ith block with B, yielding a modified ith block, setting B to the

modified ith block, and scrambling the modified ith block using at least one next round of a cipher, figure 2, page 8, second paragraph. "i" is decremented by unity and the logic continues the loops until it is determined that a predetermined number of iterations have been executed, and if not, a next round of the cipher, otherwise an encrypted stream of data blocks is output, figure 2, page 8, third paragraph continuing to page 9, end of first full paragraph; see also the pseudocode starting on page 9.

In contrast, Claim 11 recites a method for generating a tamper resistant version of a software program including a stream of data blocks that includes providing a cipher defining rounds, figure 2, page 7, iterating through the rounds of the cipher by iterating through respective outer loops of forward plain text chaining followed by backward plain text chaining, figure 2, pages 7-9, and, during each forward portion of an outer loop, applying a respective round of the cipher to each block, and during each backward portion of an outer loop, applying a respective round of the cipher to each block, id.

Claim 15 recites the inverse process disclosed on page 9, second full paragraph and accompanying pseudocode, namely, receiving a sequence of N blocks and for i= N to 1, executing a DO loop that includes reverse XORing an i^{th} block with a $block_{i-1}$. The decryption also includes unscrambling the i^{th} block using a round of a cipher to render an unscrambled block and determining whether a $block_{i-1}$ exists, and if not, executing a DO loop below, id. Otherwise, the logic includes decrementing "i" by unity and returning. The loop referred to above includes for i= 1 to N, executing a DO loop comprising reverse XORing an i^{th} block with a $block_{i+1}$, unscrambling the i^{th} block using a single round of a cipher to render an unscrambled block, and determining whether a $block_{i+1}$ exists, and if not, determining whether a predetermined number of iterations have been executed, id. Otherwise, "i" is incremented by unity.

Analysis

The issue is simple so Applicant will keep it short. In Claim 6, a data block is chained to a *plain text* version of an adjacent block in the stream to render a chained block, which is then scrambled using a first round of a cipher to render a scrambled block. The scrambling (using, recall, a *plain text* version of a block) and chaining are iterated using subsequent rounds of the cipher.

This is not what happens in Shimizu et al. Specifically, as disclosed on col. 7, line 66 continuing to col. 8, line 14 and shown best in Figure 3 of Shimizu et al., an i^{th} block is transformed (by a function F31) and XORed with a plain text block 24, with the combination then being scrambled with a key ("K2") by means of a function F32. So far, so good as relates to Claim 6.

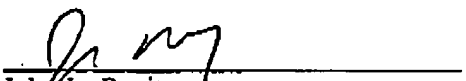
But then Shimizu et al. diverges from claim 6 in that it does not iterate between ***plain text*** chaining and XORing. Instead, as shown in Figure 3 and disclosed at col. 8, lines 5-10, the scrambled result is not XORed with another plain text block as required by Claim 6, but rather is XORed with the ***scrambled*** block that had been output by the function F31. This result is then XOR'ed with the ***scrambled*** result of the function F32, and so on, i.e., in the iterative process of Shimizu et al. chaining of successive ***plain text*** blocks is not alternated with scramblings, but instead two and only two blocks are chained and then scrambled N times before any further plain text blocks are implicated. This is emphasized in Figures 2 and 5 of Shimizu et al. and explained at col. 8, lines 42-58. It appears from col. 8, lines 19 and 20 that Shimizu et al. regards the above-described processing as "Feistel-type" encryption, which is precisely the kind of fault-tolerant encryption method sought to be avoided herein, see the present background.

With respect to Claim 11, for the reasons above it is clear that Shimizu et al. does not iterate through the rounds of the cipher by iterating through respective outer loops of forward plain text chaining followed

by backward plain text chaining, while, during each forward portion of an outer loop, applying a respective round of the cipher to each block.

The specific limitations of Claims 8 and 15 have not been identified in the prior art as is otherwise required to establish a *prima facie* case of obviousness, see MPEP §2143. Accordingly, on its face the rejection falls. Further, to simply wave off a series of specifically claimed algorithm steps as "not departing from the spirit and scope of Shimizu et al." reflects a profound misunderstanding of the law of patentability. There simply is no requirement to depart from the "spirit and scope" of the prior art, whatever that might mean. Rather, unless all limitations are shown by evidence of record as either being in the prior art or within the level of skill of the art (and this means beyond a mere allegation unsupported by evidence of record that something is in the general knowledge of the art), a claim is patentable.

Respectfully submitted,



John L. Rogitz
Registration No. 33,549
Attorney of Record
750 B Street, Suite 3120
San Diego, CA 92101
Telephone: (619) 338-8075

JLR:jg

1053-99.AM3